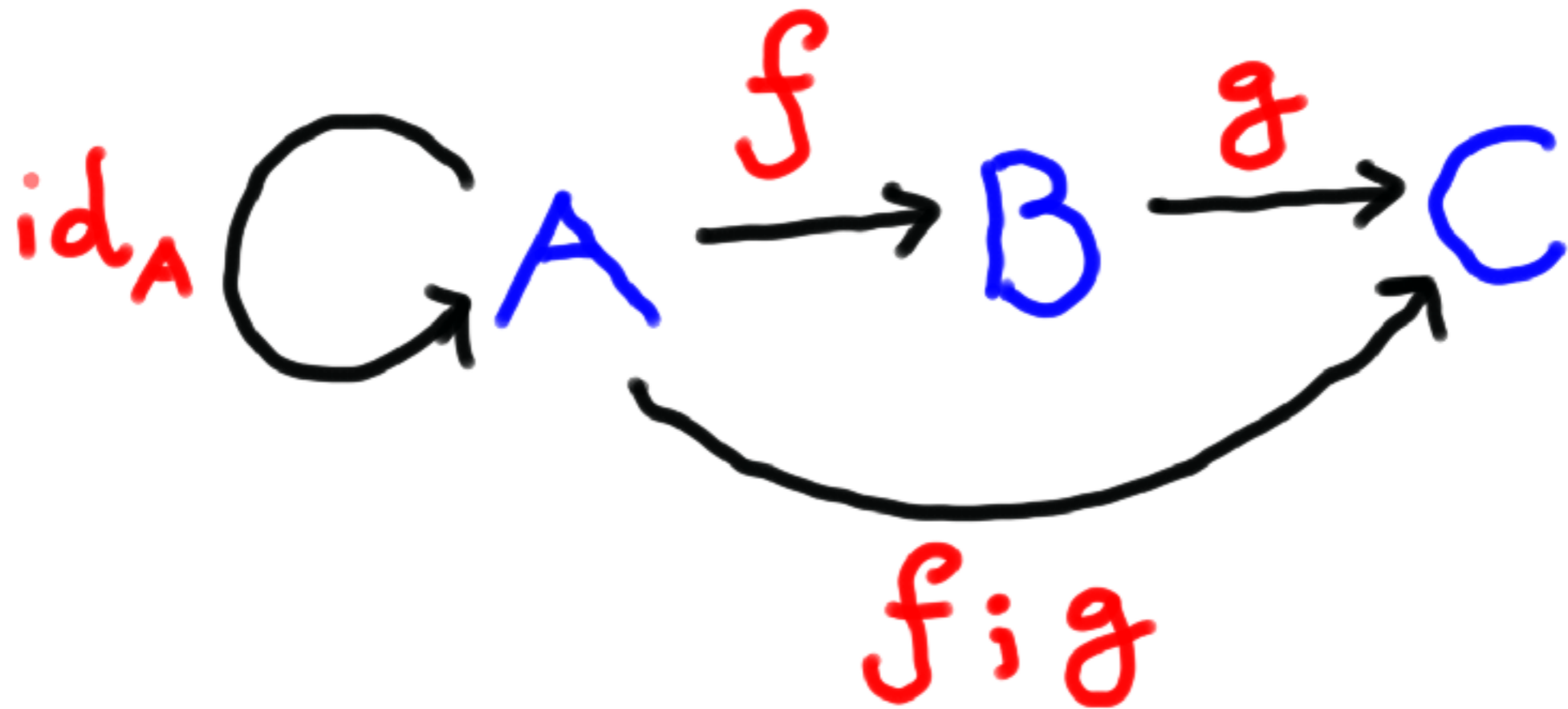


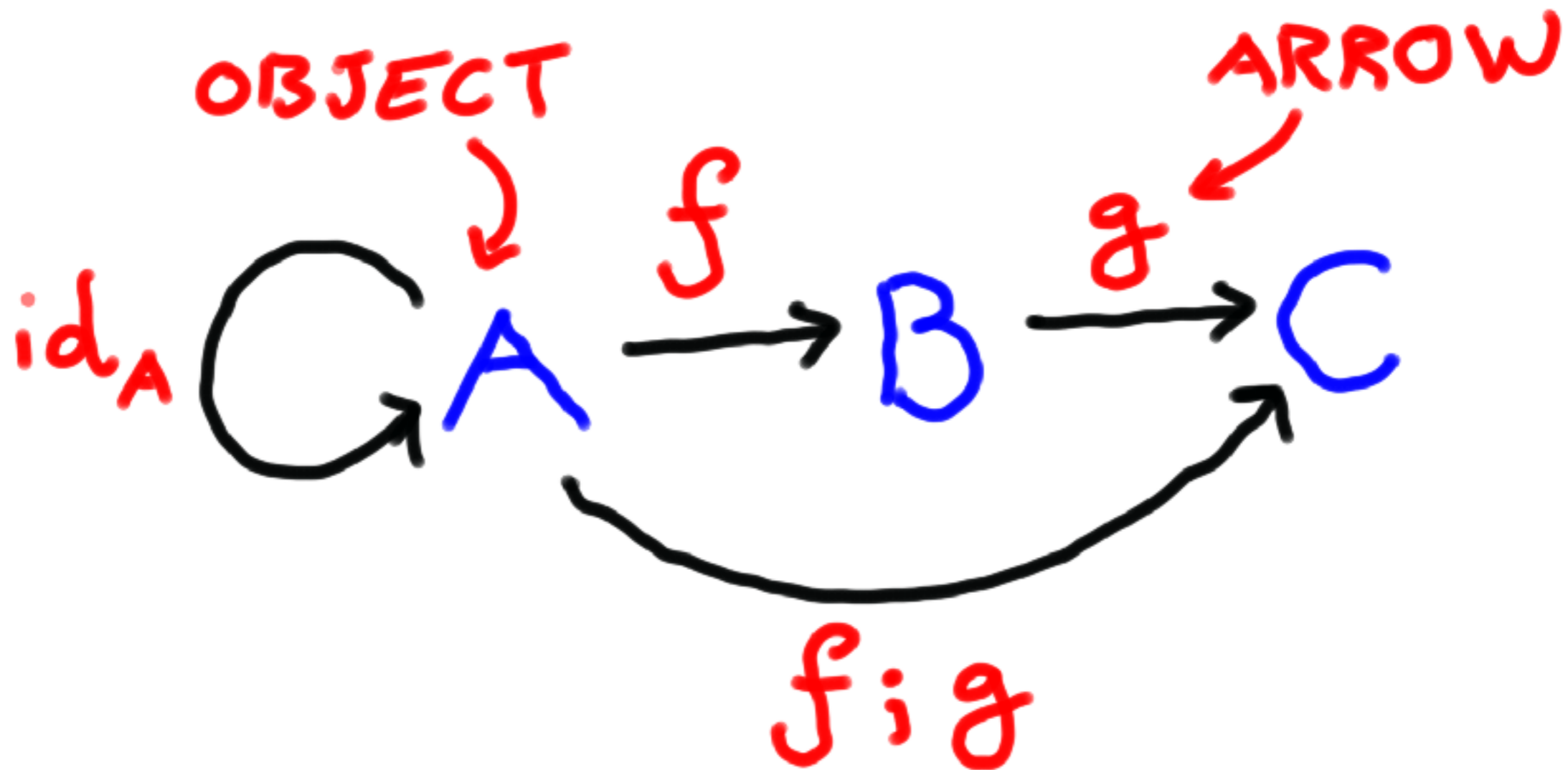
# Categories for the Working Hacker

Philip Wadler  
University of Edinburgh  
QCon SF, 15 Nov 2017

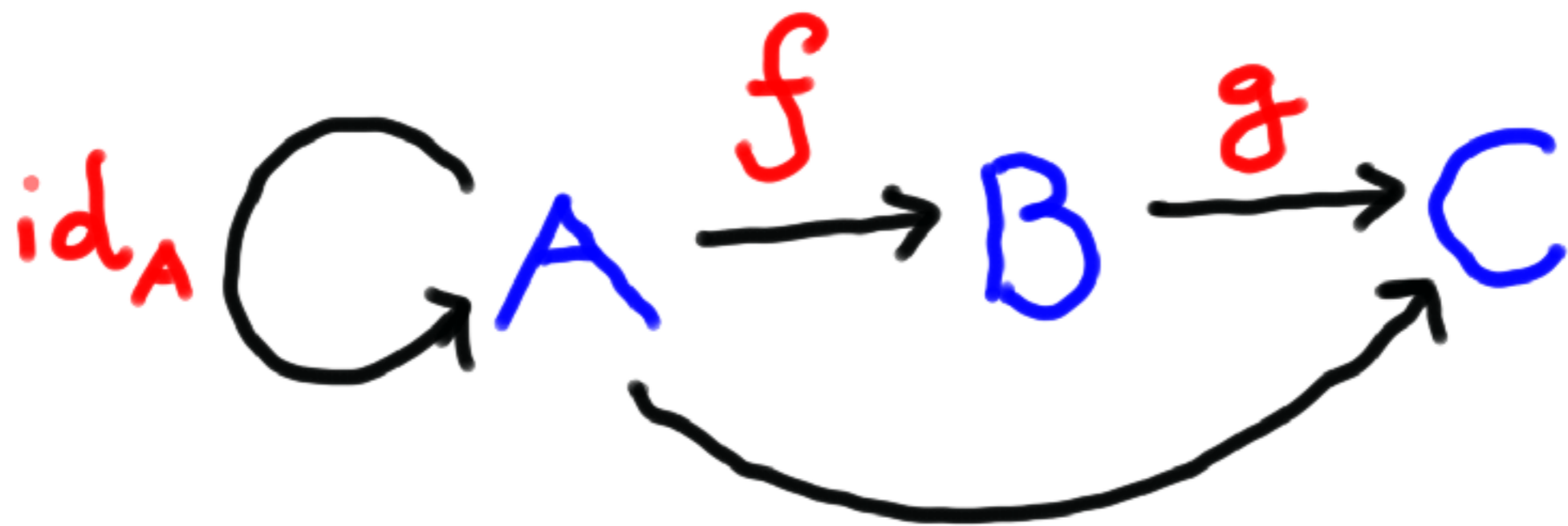
# CATEGORIES



# CATEGORIES

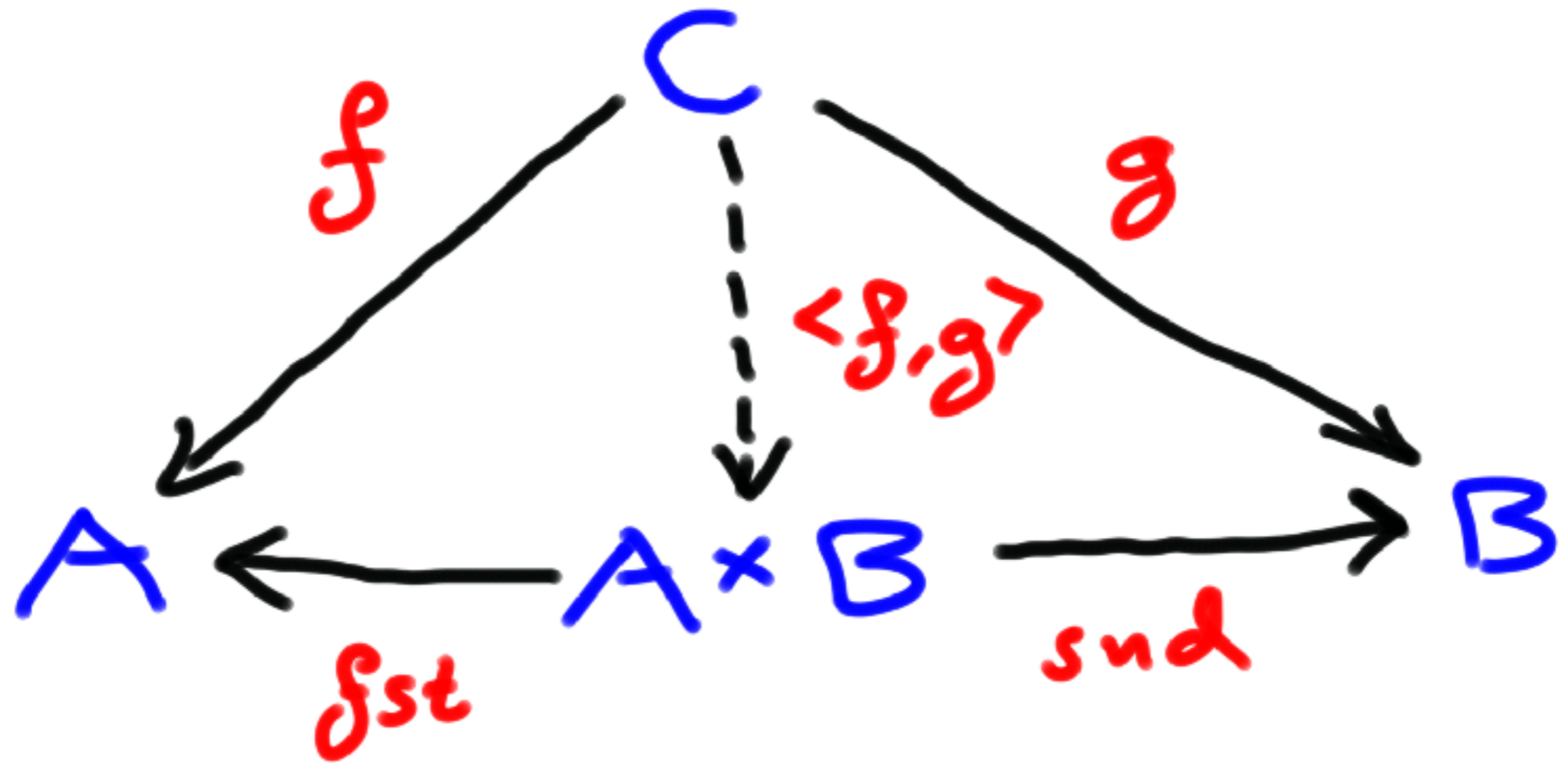


# CATEGORIES

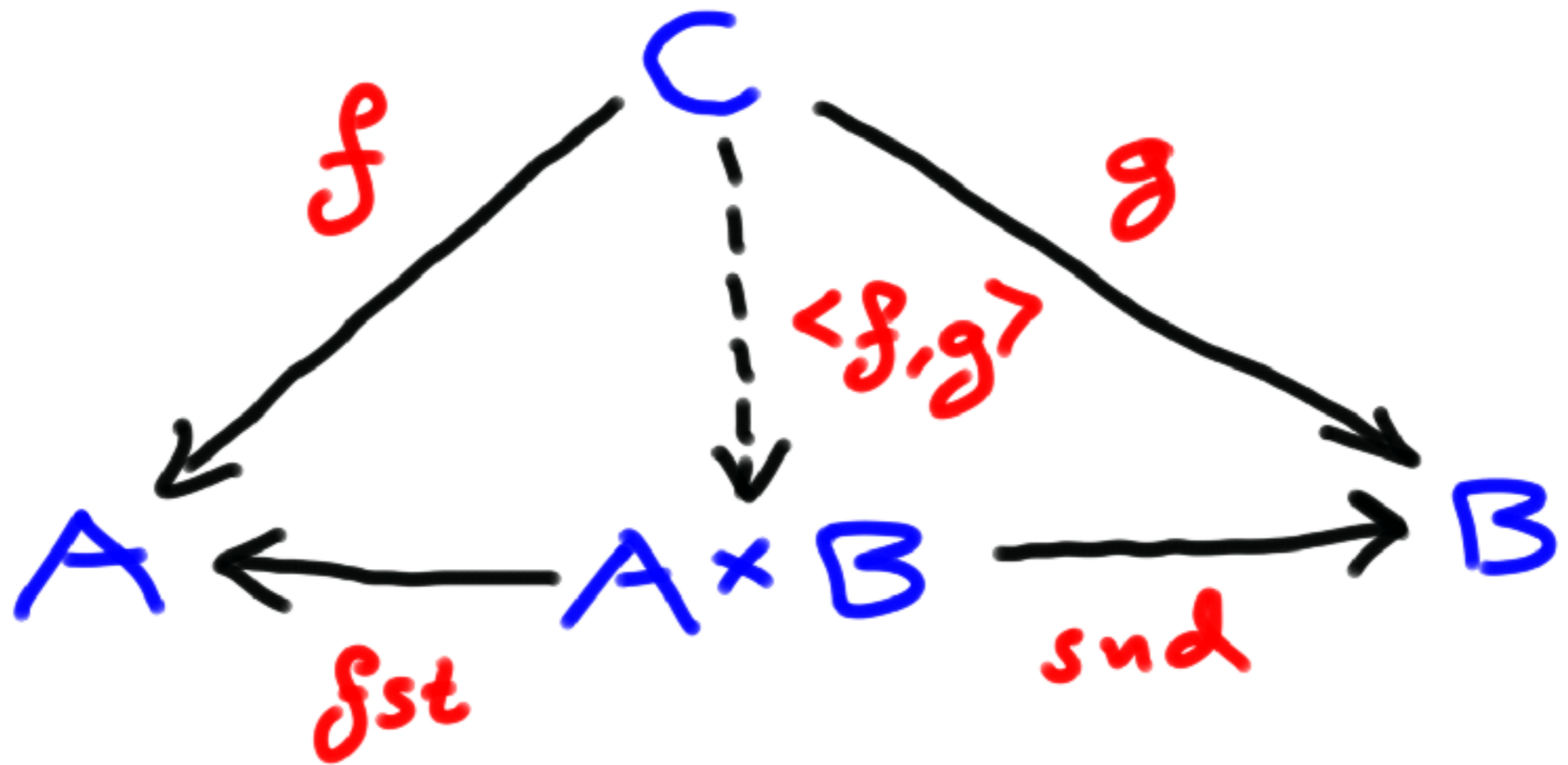


$$id_A \circ f = f = f \circ id_B$$
$$(f \circ g) \circ h = f \circ (g \circ h)$$

# PRODUCTS



# PRODUCTS



$$\langle f, g \rangle; \text{fst} = f$$
$$\langle f, g \rangle; \text{snd} = g$$

# PRODUCTS

$$3 \times 2$$

$(a, 0)$

$(a, 1)$

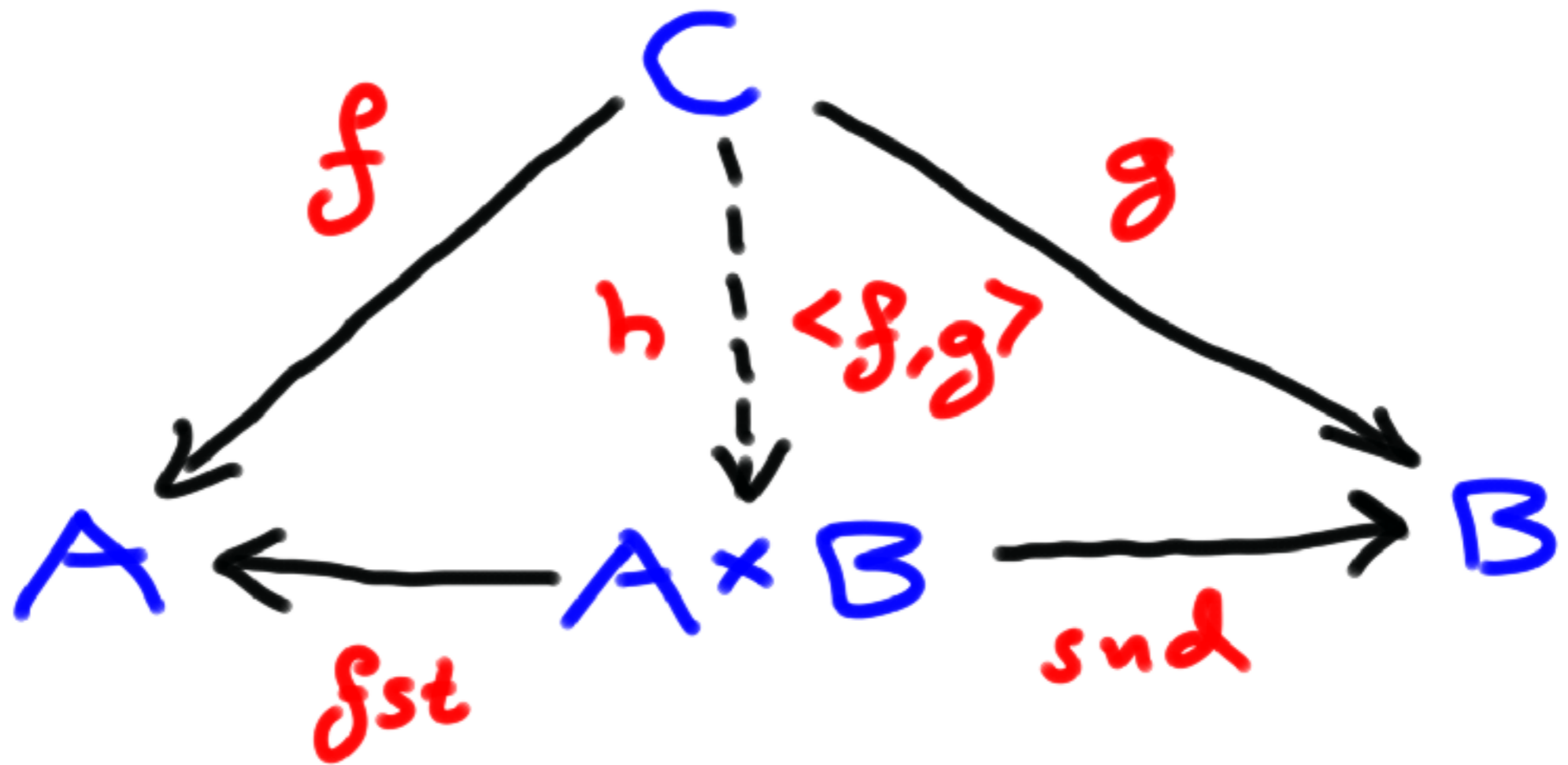
$(b, 0)$

$(b, 1)$

$(c, 0)$

$(c, 1)$

# PRODUCTS



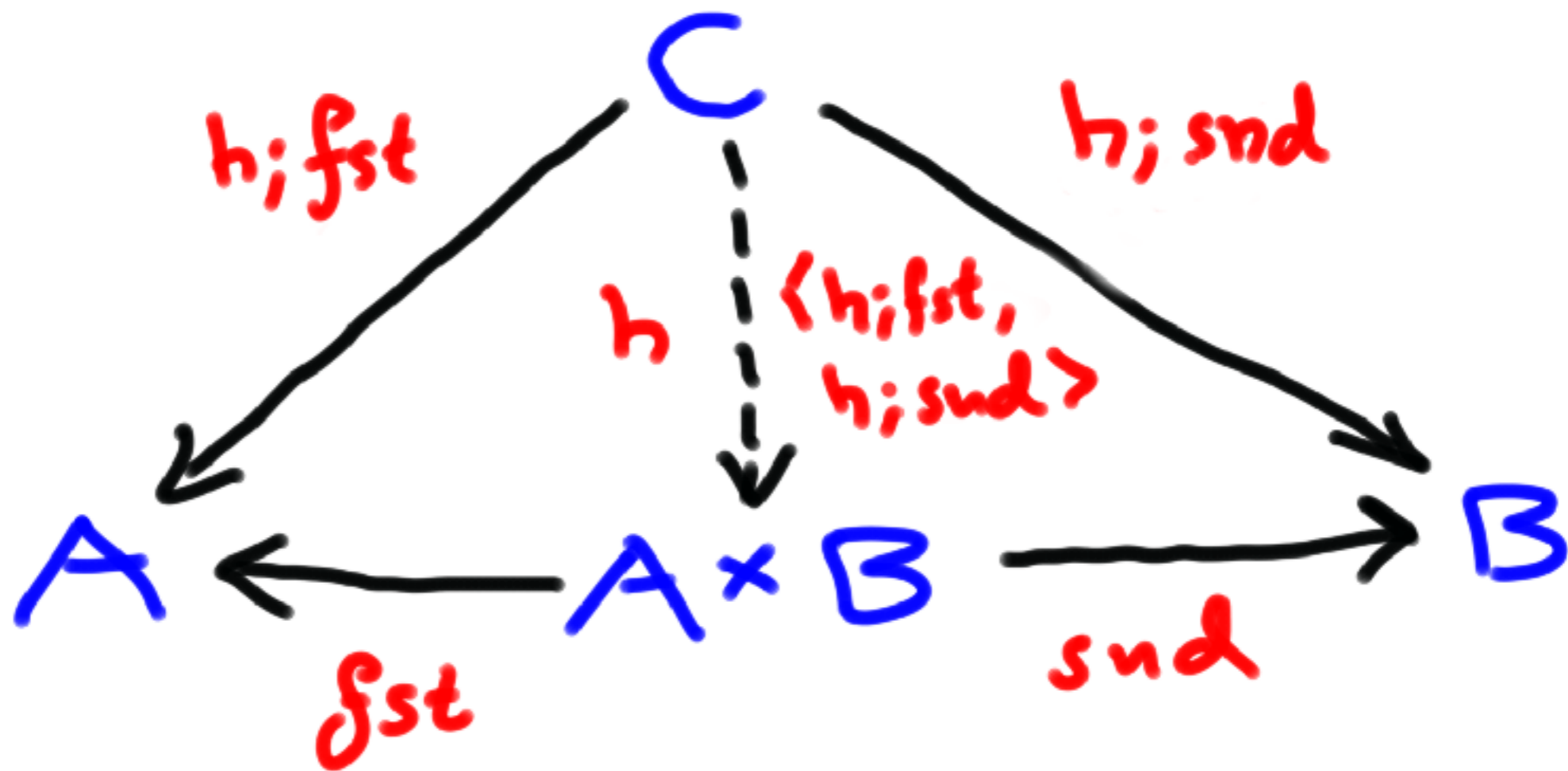
$$h; \text{fst} = f \quad h; \text{snd} = g$$

---

$$h = \langle f, g \rangle$$



# PRODUCTS



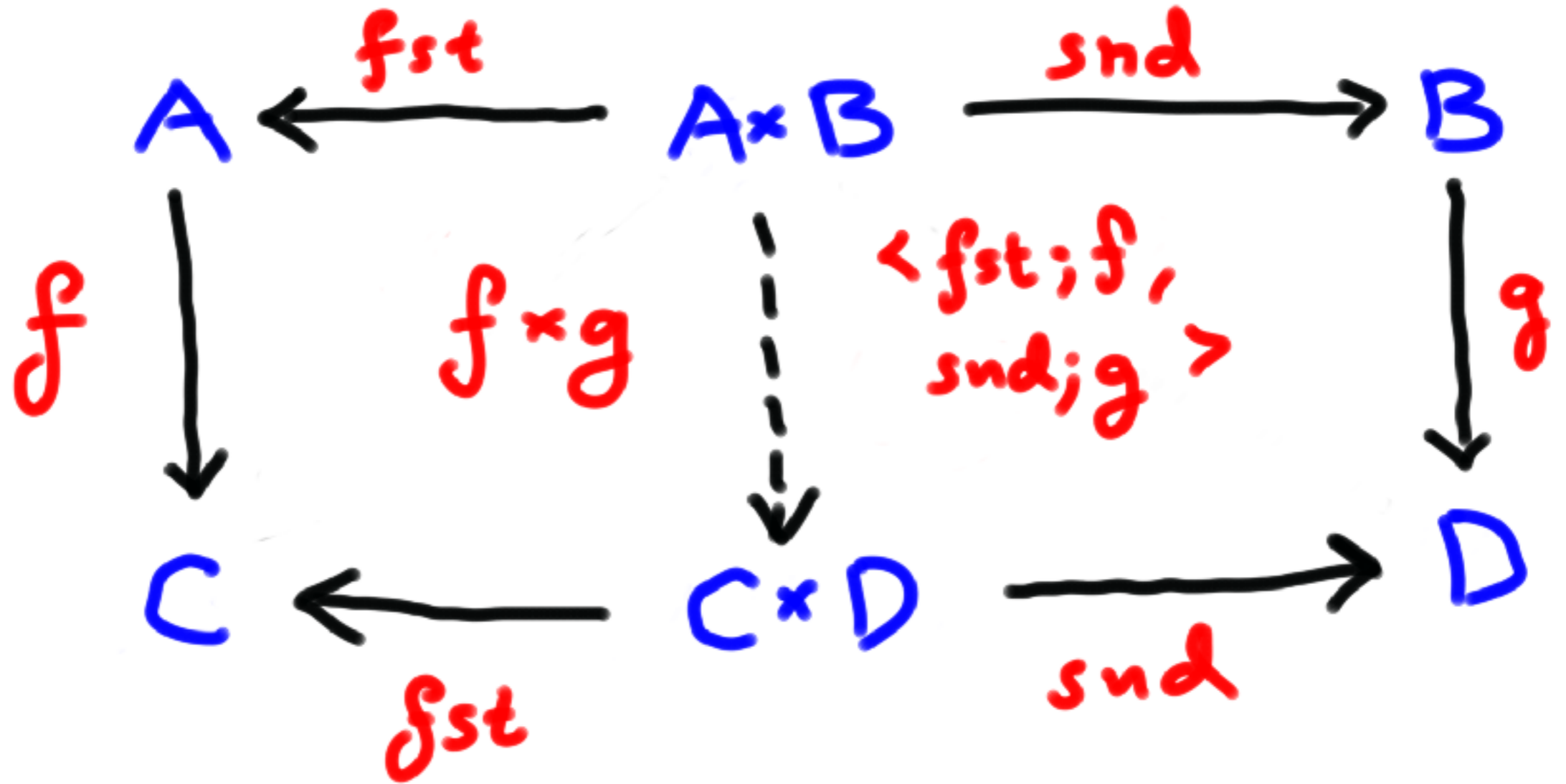
$$h;fst = h;fst$$

$$h;snd = h;snd$$

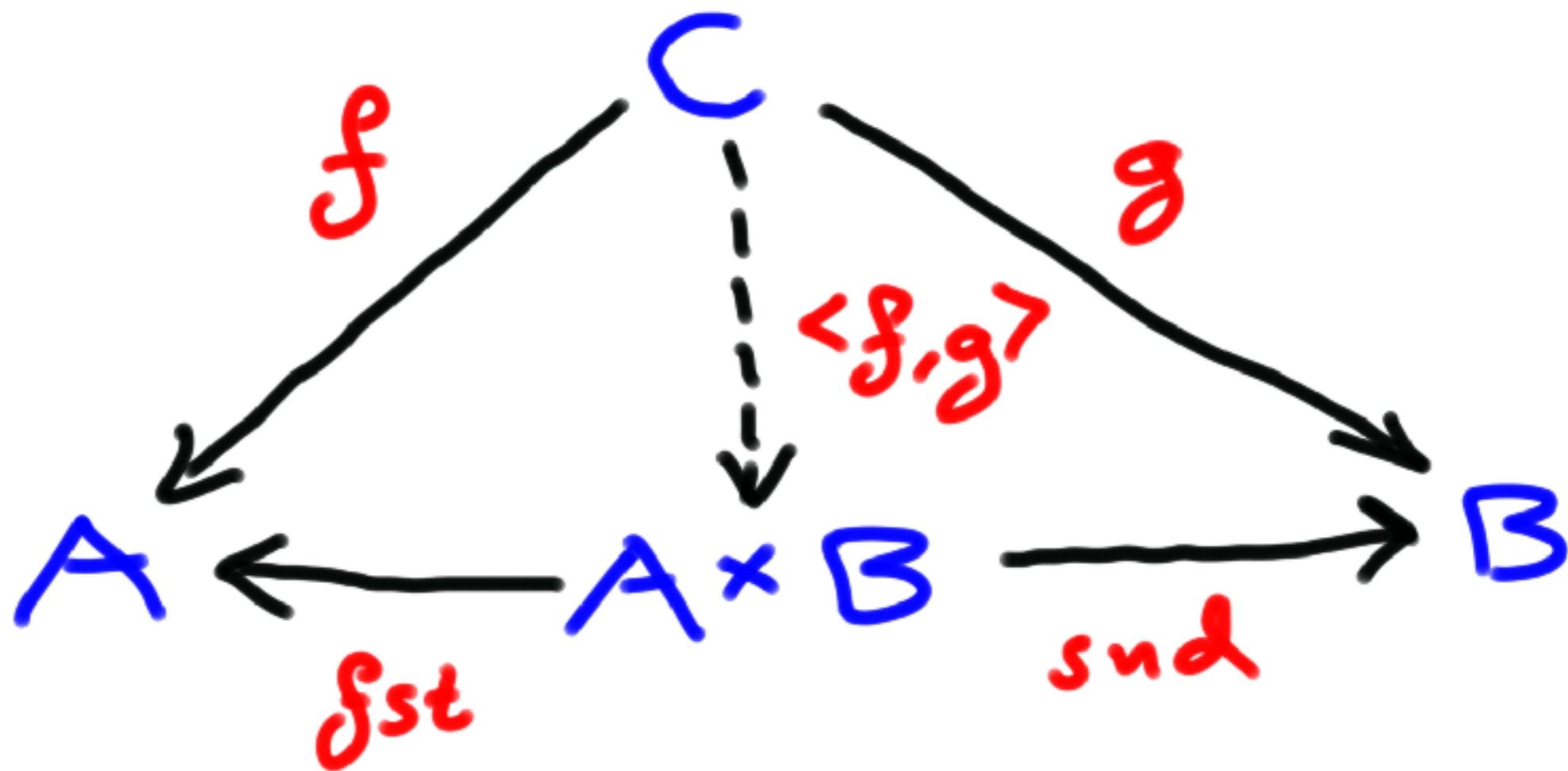
---

$$h = \langle h;fst, h;snd \rangle$$

# PRODUCTS



# PRODUCTS



$$\mathcal{C}(C, A \times B) \cong \mathcal{C}(C, A) \times \mathcal{C}(C, B)$$

# Products in Java

```
public class Product<A,B> {  
    private A fst;  
    private B snd;  
    public Product(A fst, B snd) {  
        this.fst = fst; this.snd = snd;  
    }  
    public A getFst() {  
        return this.fst;  
    }  
    public B getSnd() {  
        return this.snd;  
    }  
}
```

# Products in Java

```
public class Test {  
    public Product<Integer,String> pair =  
        new Product(1, "two");  
    public Integer one = pair.getFirst();  
    public String two = pair.getSecond();  
}
```

# Products in Haskell

```
data Product a b =  
  Pair { fst :: a, snd :: b }
```

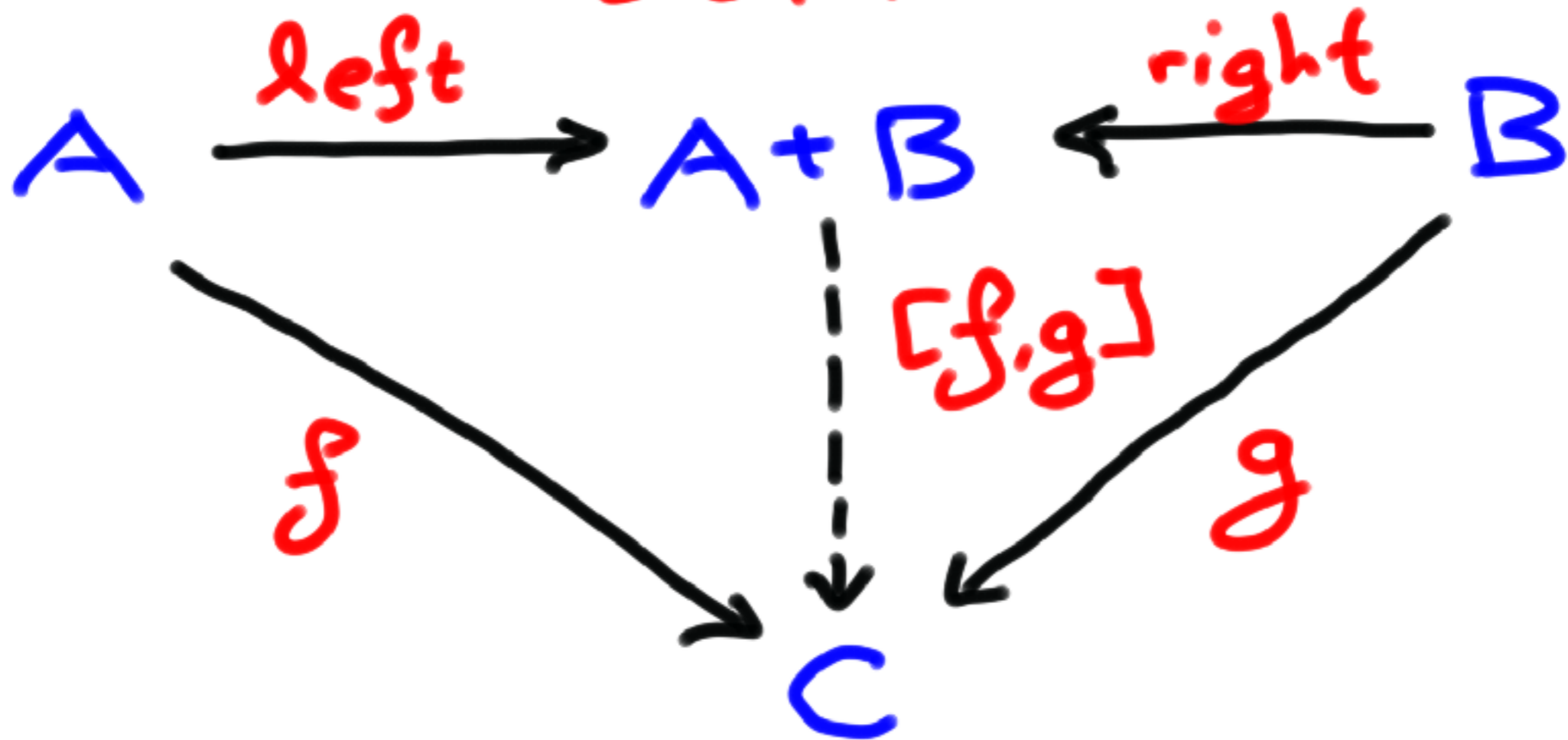
# Products in Haskell

```
pair :: Product Int String  
pair = Pair 1 "two"
```

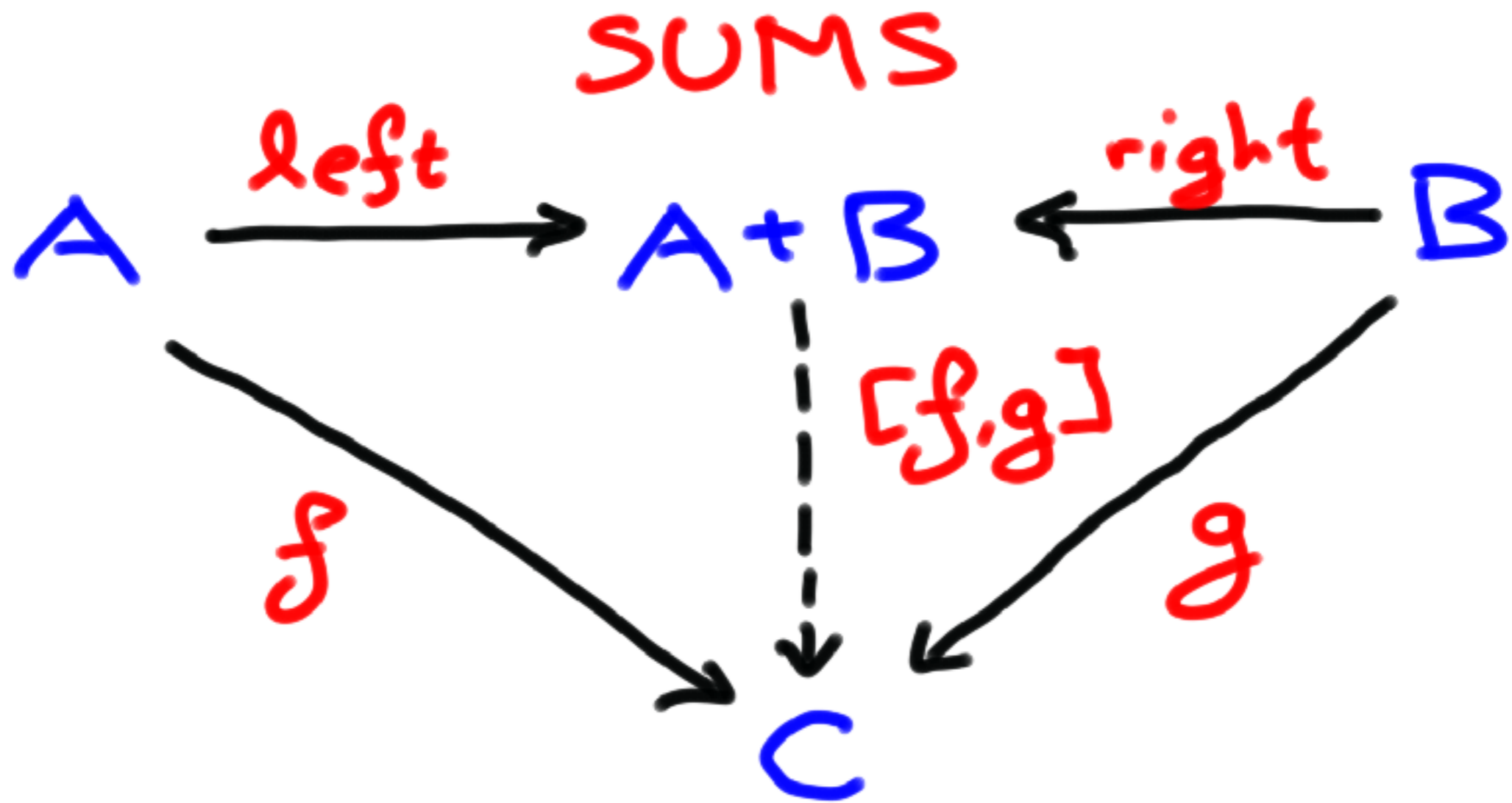
```
one :: Int  
one = fst pair
```

```
two :: String  
two = snd pair
```

# SUMS







left;  $[f, g] = f$   
right;  $[f, g] = g$

SUMS

$$3+2$$

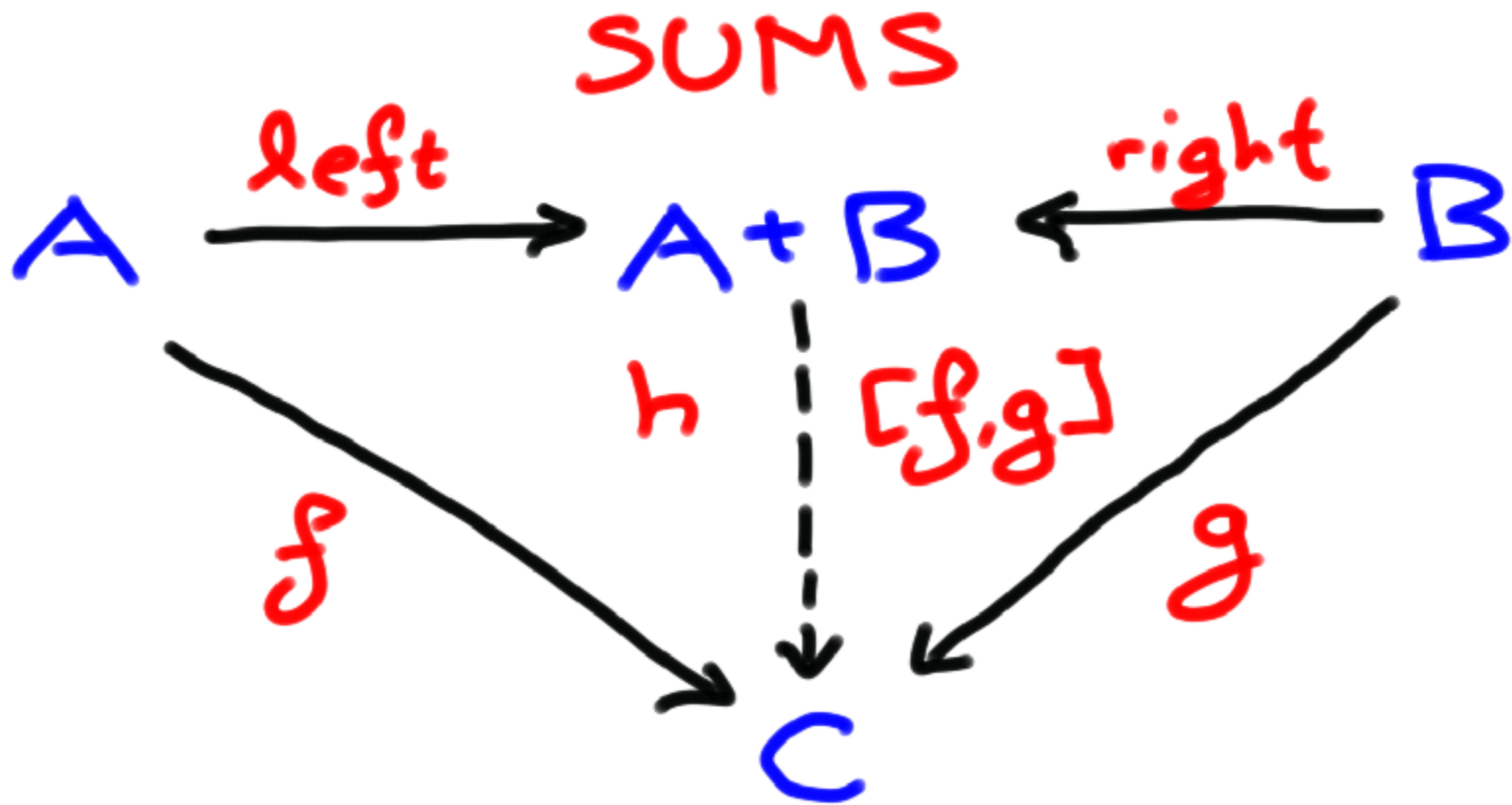
left 'a'

right 0

left 'b'

right 1

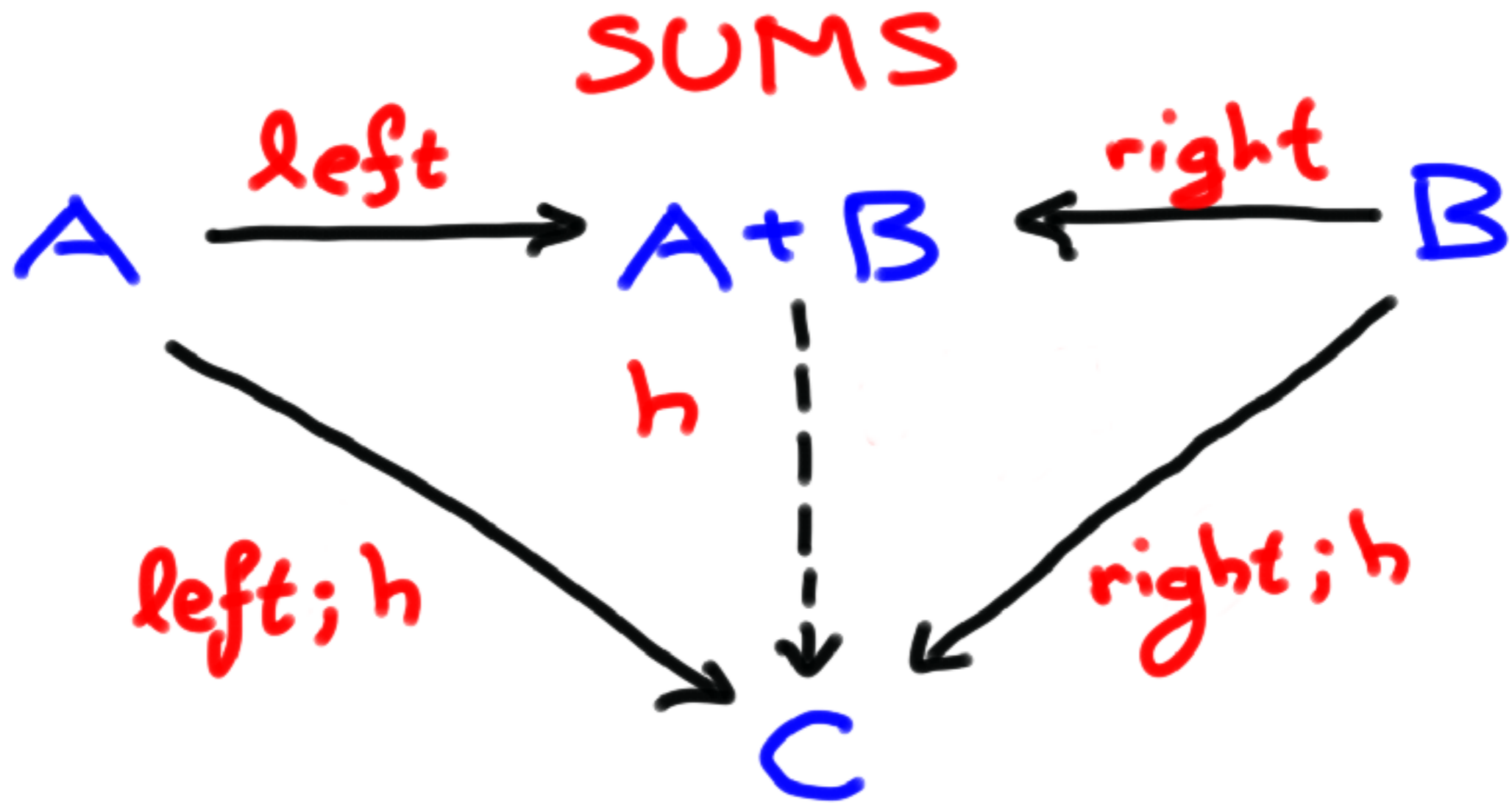
left 'c'



$left; h = f$        $right; h = g$   

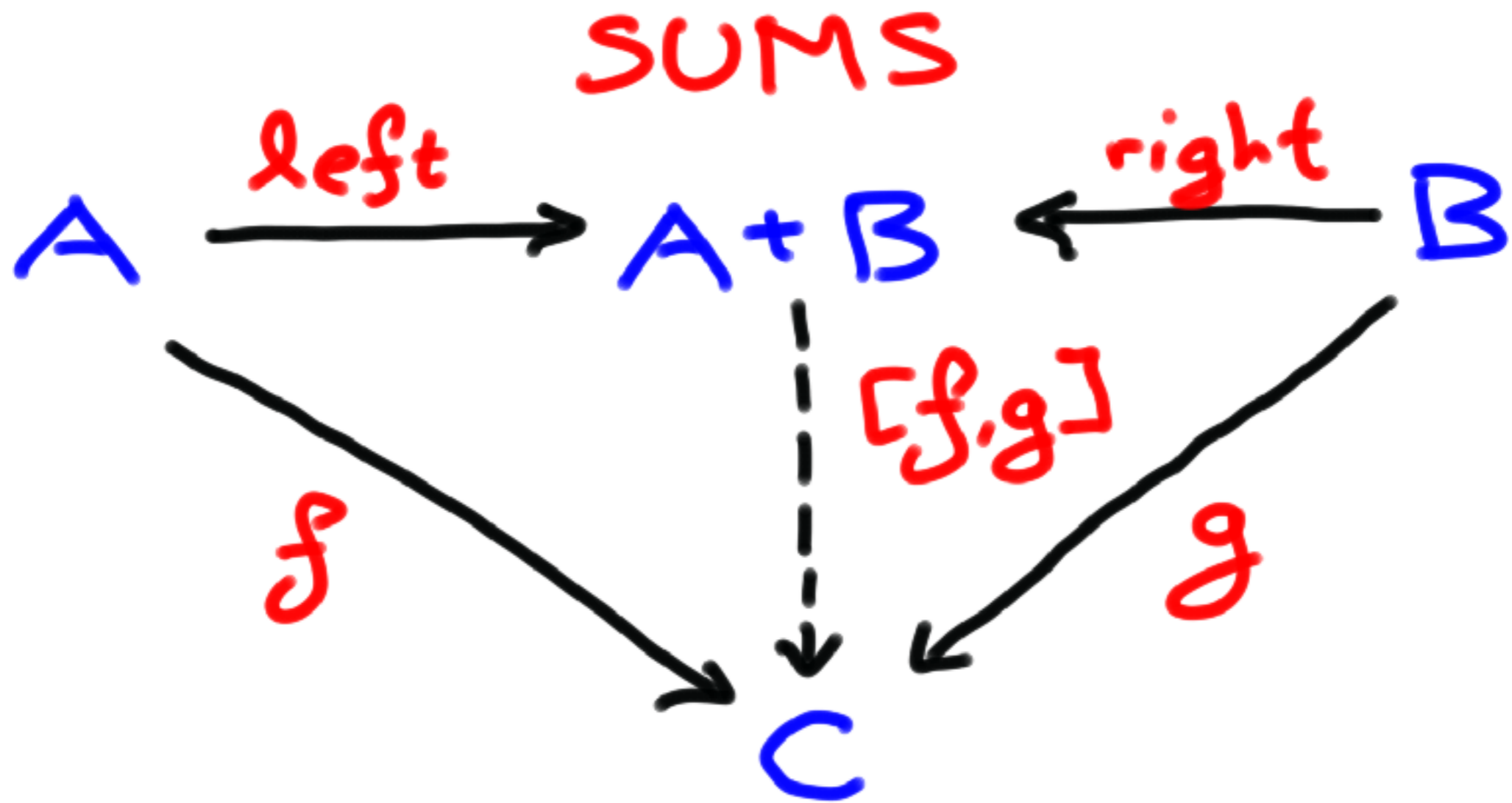

---

 $h = [f, g]$



$left; h = left; h$        $right; h = right; h$   

 $h = [left; h, right; h]$



$$\mathcal{C}(A+B, C) \cong \mathcal{C}(A, C) \times \mathcal{C}(B, C)$$

# Sums in Java

```
public interface Sum<A,B> {
    public <C> C caseExpr(Function<A,C> f,
                        Function<B,C> g);
}
public class Left<A,B> implements Sum<A,B> {
    private A x;
    public Left(A x) { this.x = x; }
    public <C> C caseExpr(Function<A,C> f,
                        Function<B,C> g) {
        return f.apply(x);
    }
}
public class Right<A,B> implements Sum<A,B> {
    private B y;
    public Right(B y) { this.y = y; }
    public <C> C caseExpr(Function<A,C> f,
                        Function<B,C> g) {
        return g.apply(y);
    }
}
```

# Sums in Java

```
public class ErrInt extends Sum<String,Integer> {
    public ErrInt err = new Left("error");
    public ErrInt one = new Right(1);
    public ErrInt add(ErrInt that) {
        return this.caseExpr(
            e -> new Left(e),
            m -> that.caseExpr(
                e -> new Left(e),
                n -> new Right(m+n)
            )
        );
    }
    public ErrInt test = one.add(err);
}
```

# Sums in Haskell

```
data Sum a b = Left a | Right b
```



# Sums in Haskell

```
type ErrInt = Sum String Int
```

```
err = Left "error"
```

```
one = Right 1
```

```
add :: ErrInt -> ErrInt -> ErrInt
```

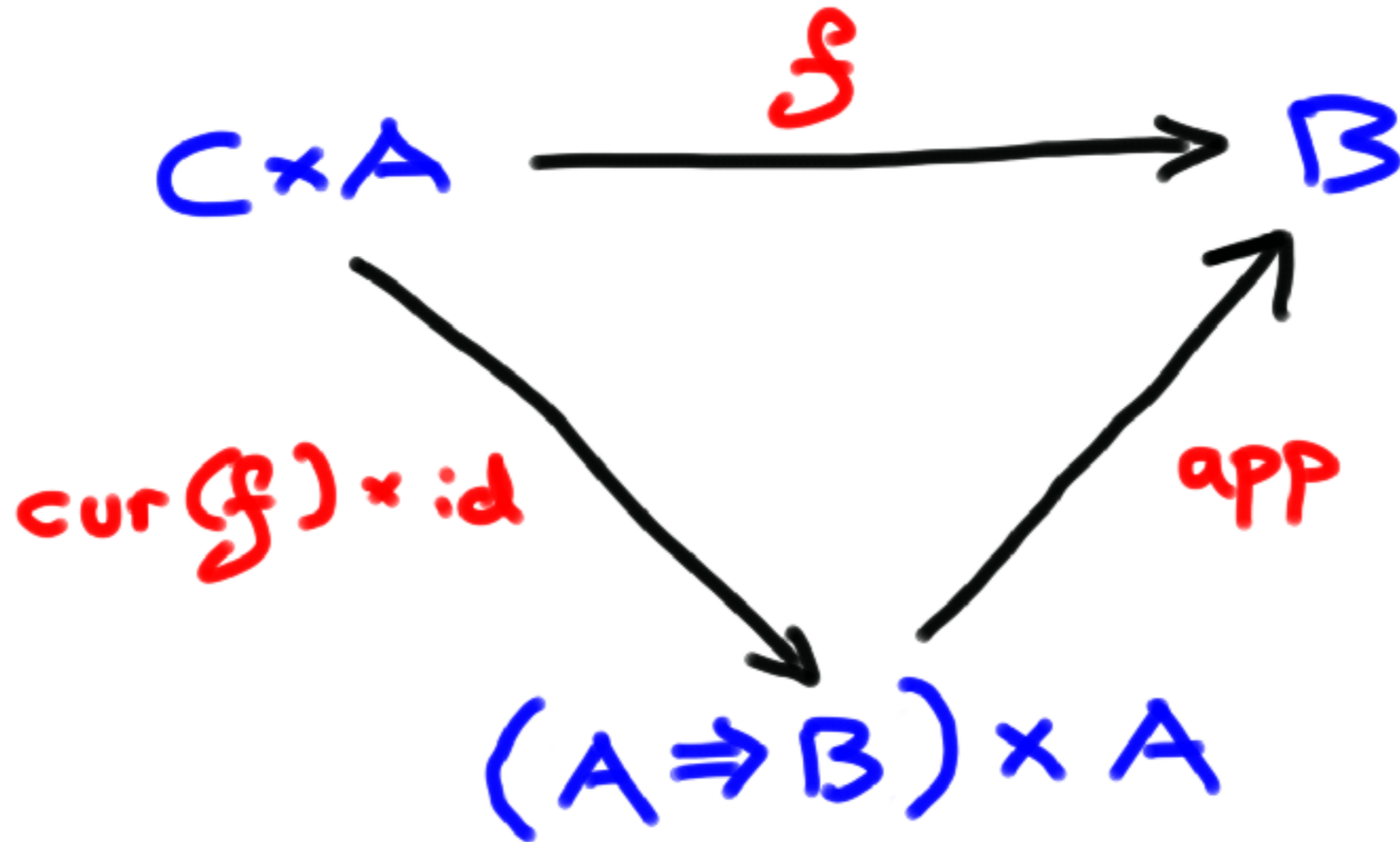
```
add (Left e) that = Left e
```

```
add this (Left e) = Left e
```

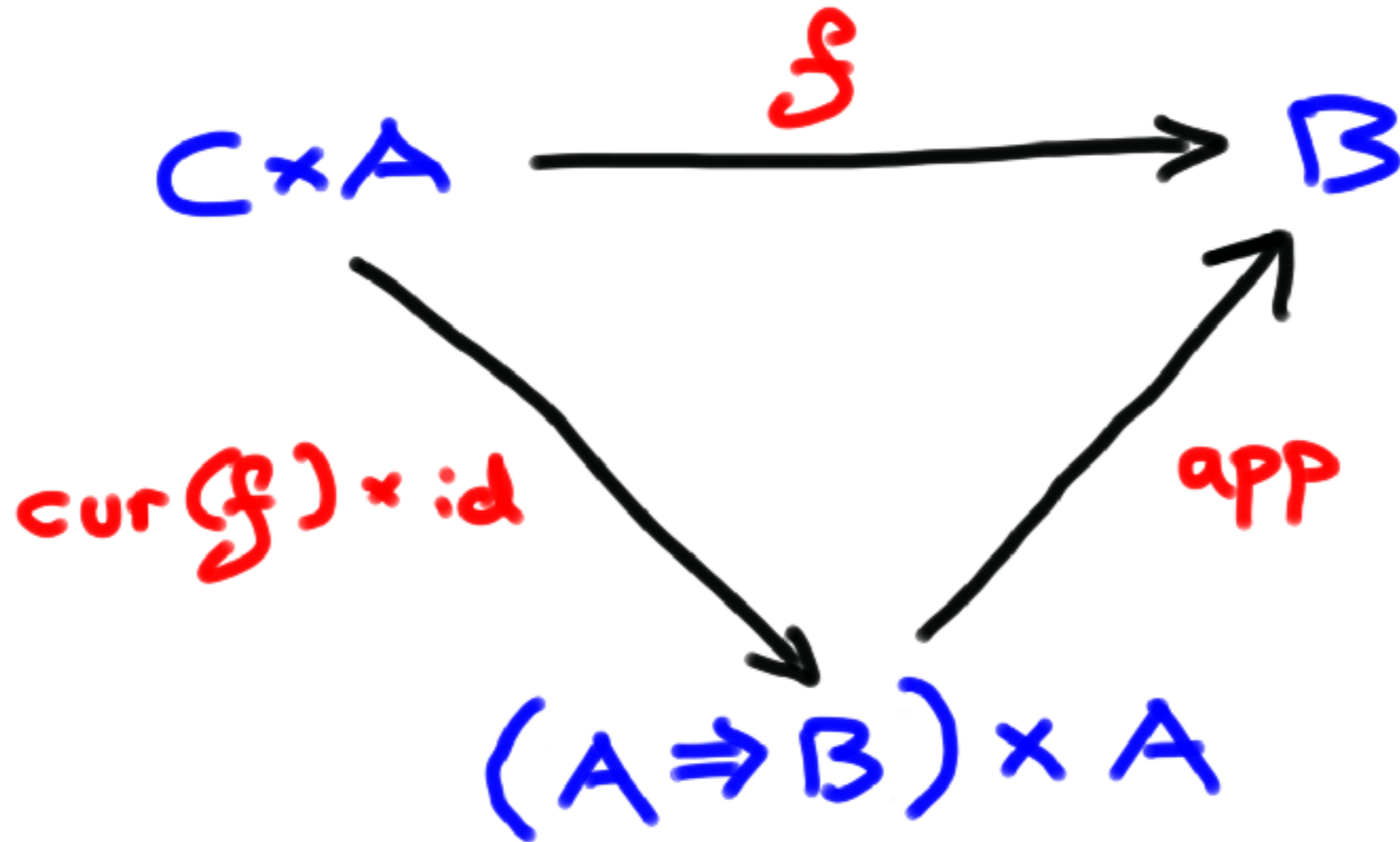
```
add (Right m) (Right n) = Right (m+n)
```

```
test = add one err
```

# EXPONENTIALS



# EXPONENTIALS



$$(\text{cur}(f) \times \text{id}) ; \text{app} = f$$

# EXPONENTIALS

$$2 \Rightarrow 3 = 3^2$$

$$\begin{array}{l} 0 \mapsto 'a' \\ 1 \mapsto 'a' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'b' \\ 1 \mapsto 'a' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'c' \\ 1 \mapsto 'a' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'a' \\ 1 \mapsto 'b' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'b' \\ 1 \mapsto 'b' \end{array}$$

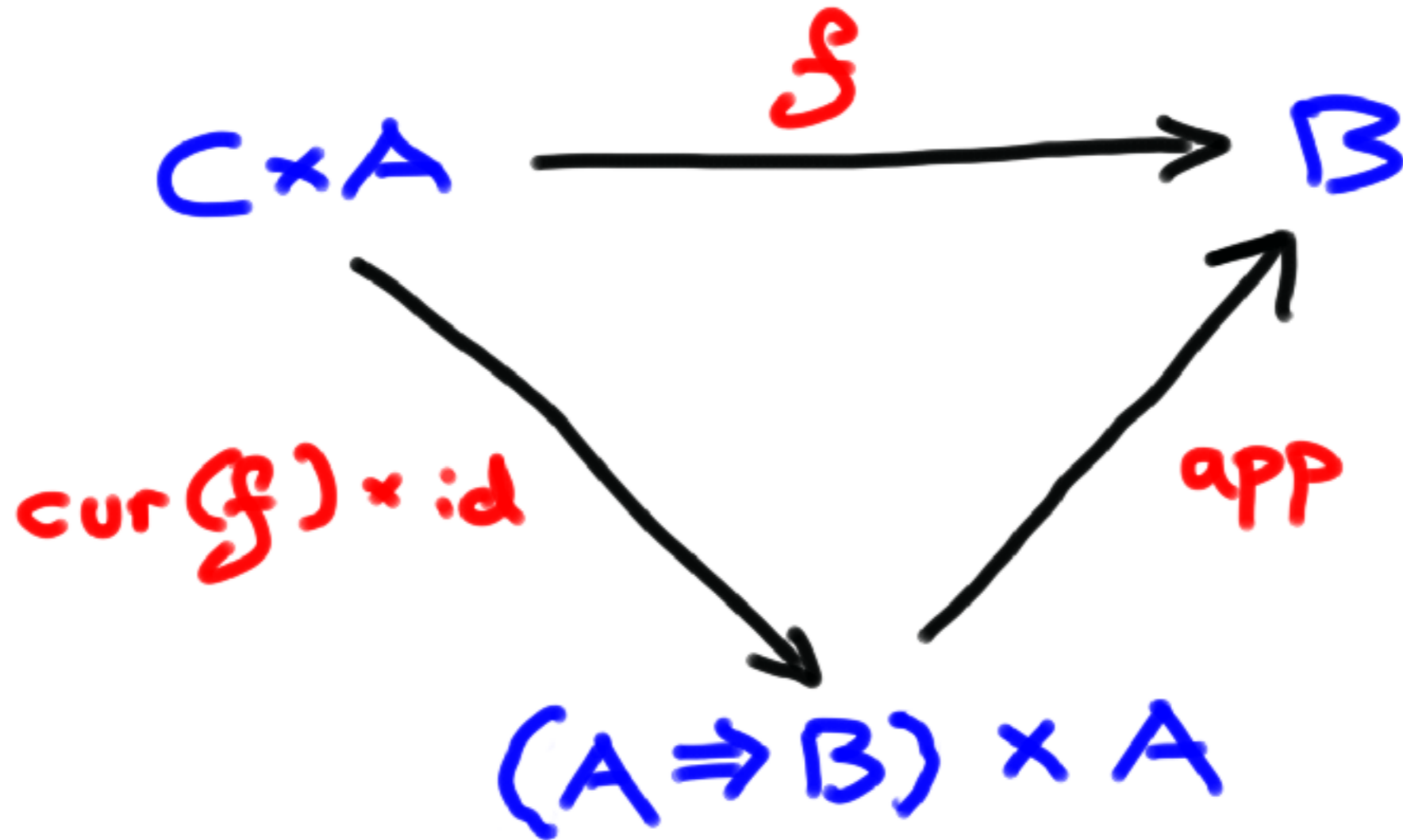
$$\begin{array}{l} 0 \mapsto 'c' \\ 1 \mapsto 'b' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'a' \\ 1 \mapsto 'c' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'b' \\ 1 \mapsto 'c' \end{array}$$

$$\begin{array}{l} 0 \mapsto 'c' \\ 1 \mapsto 'c' \end{array}$$

# EXPONENTIALS



$$\mathcal{C}(C \times A, B) \cong \mathcal{C}(C, [A \Rightarrow B])$$

# Exponentials in Java

```
public class Test {  
    public Function<Integer,Integer>  
        add (Integer n) {  
        return x -> x + n;  
        }  
    public Function<Integer,Integer> incr =  
        add(1);  
    public Integer three = incr.apply(2);  
}
```

# Exponentials in Haskell

```
add :: Int -> (Int -> Int)
add n = \x -> n + x
```

```
incr :: Int -> Int
incr = add 1
```

```
three :: Int
three = incr 2
```

# Exponentials in Haskell

```
add :: Int -> Int -> Int
```

```
add n x = n + x
```

```
incr :: Int -> Int
```

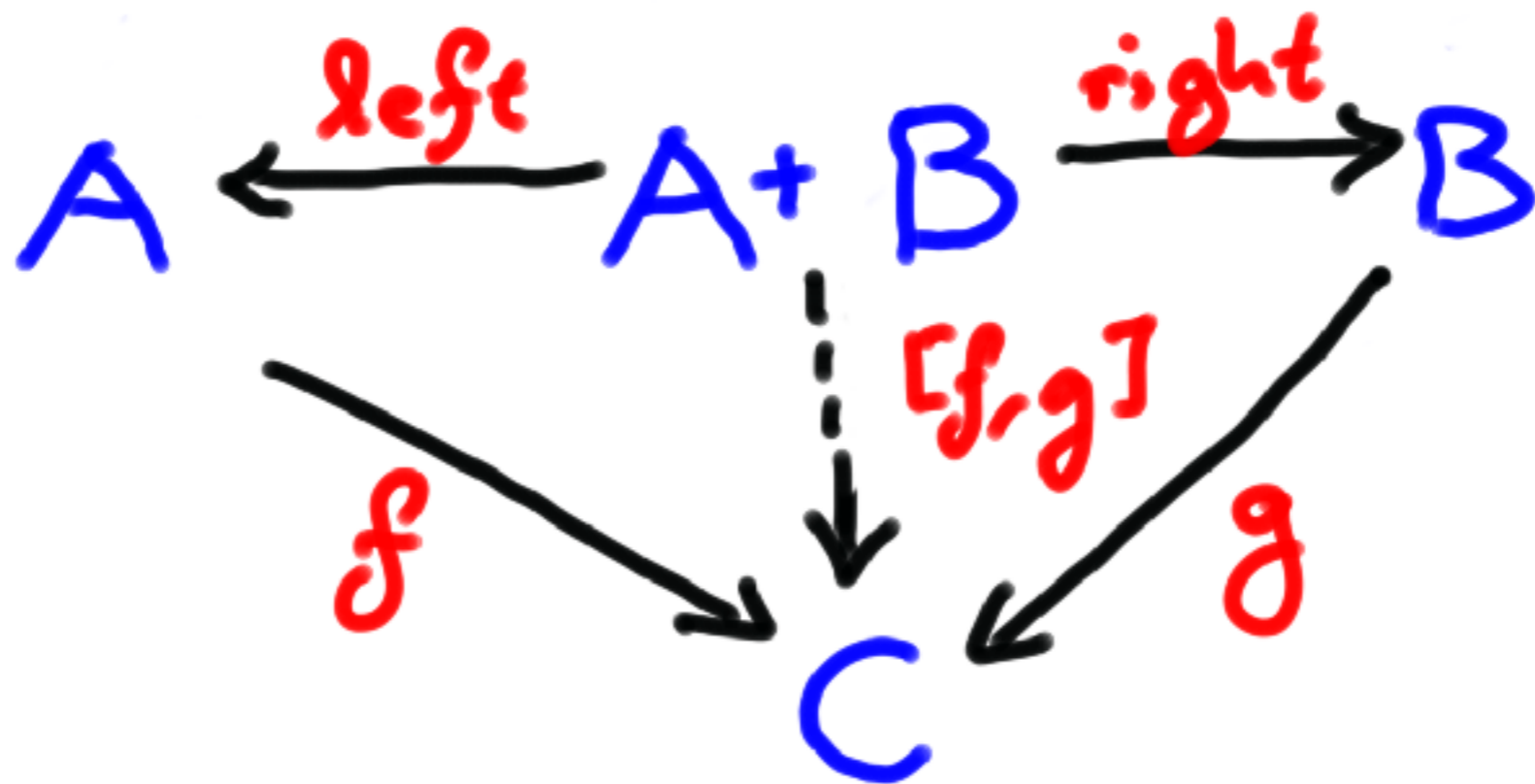
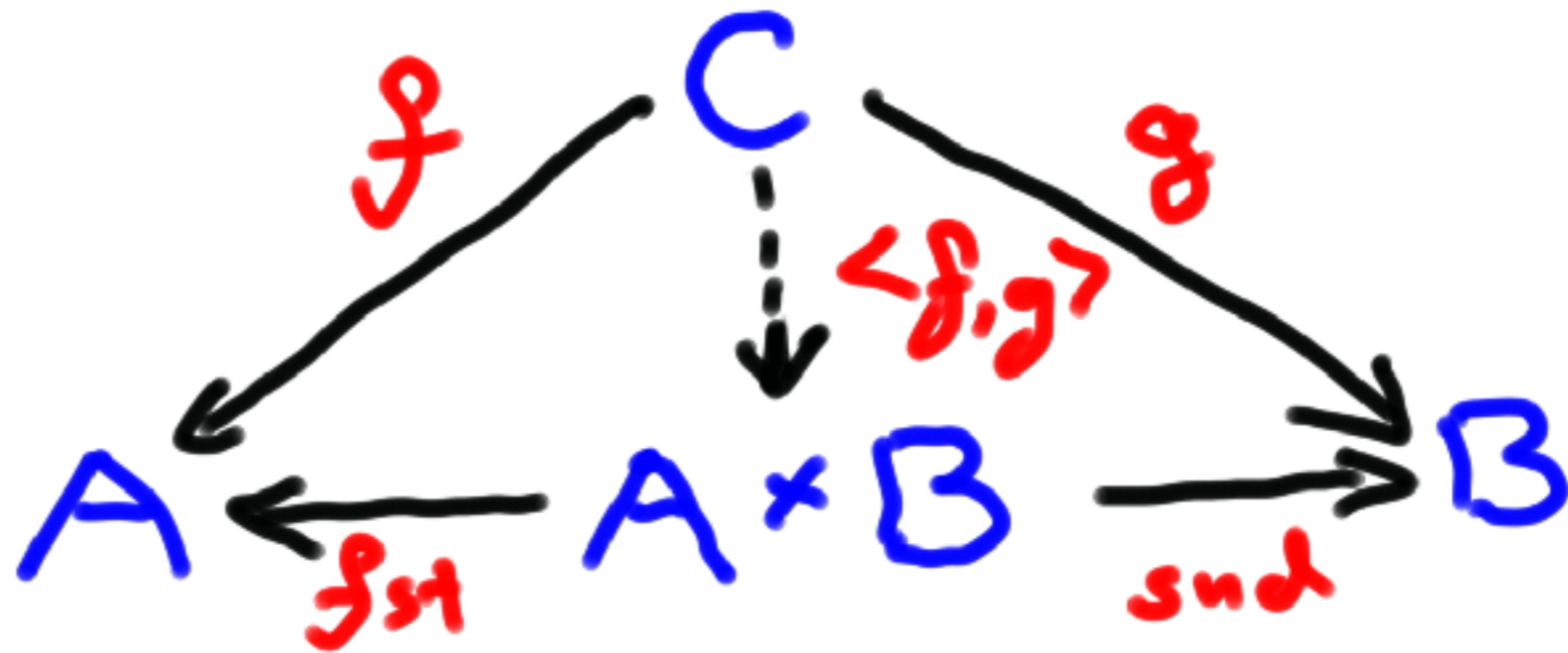
```
incr = add 1
```

```
three :: Int
```

```
three = incr 2
```



# DUALS



# ISOMORPHISMS

$$\mathcal{C}(C, A \times B) \cong \mathcal{C}(C, A) \times \mathcal{C}(C, B)$$

$$\mathcal{C}(A + B, C) \cong \mathcal{C}(A, C) \times \mathcal{C}(B, C)$$

$$\mathcal{C}(C, [A \Rightarrow B]) \cong \mathcal{C}(C \times A, B)$$

# HIGH SCHOOL

$$(A \times B)^C = A^C \times B^C$$

$$C^{(A+B)} = C^A \times C^B$$

$$(B^A)^C = B^{C \times A}$$

# Further Reading

- Saunders MacLane, *Categories for the Working Mathematician*
- Benjamin Pierce, *Basic Category Theory for Computer Scientists*
- Bartosz Milewski, *Programming Cafe* (blog)